Bottleneck and Embedding Representation of Speech for DNN-based Language and Speaker Recognition

Alicia Lozano-Diez, Joaquin Gonzalez-Rodriguez, Javier Gonzalez-Dominguez

Audias-UAM, Universidad Autonoma de Madrid, Madrid, Spain
alicia.lozano@uam.es

Abstract

In this manuscript, we summarize the findings presented in Alicia Lozano Diez's Ph.D. Thesis, defended on the 22nd of June, 2018 in Universidad Autonoma de Madrid (Spain). In particular, this Ph.D. Thesis explores different approaches to the tasks of language and speaker recognition, focusing on systems where deep neural networks (DNNs) become part of traditional pipelines, replacing some stages or the whole system itself. First, we present a DNN as classifier for the task of language recognition. Second, we analyze the use of DNNs for feature extraction at frame-level, the so-called bottleneck features, for both language and speaker recognition. Finally, utterance-level representation of the speech segments learned by the DNN (known as embedding) is described and presented for the task of language recognition. All these approaches provide alternatives to classical language and speaker recognition systems based on i-vectors (Total Variability modeling) over acoustic features (MFCCs, for instance). Moreover, they usually yield better results in terms of performance.

1. Introduction

Lately, automatic speech recognition (ASR) has experienced a breathtaking progress, partially thanks to the introduction of deep neural networks (DNNs) into their approaches. This has spread across related areas such as language identification (LID) and speaker recognition (SID), where DNNs have noticeably improved their performance.

In this manuscript we present a summary of the main findings of the Ph.D. Thesis defended by Alicia Lozano Díez, where we focused on different approaches for LID and SID based on DNNs, replacing some stages or the whole system. The complete dissertation can be found in [1].

First, end-to-end language recognition systems based on DNNs are analyzed, where the network is used as classifier directly. We focus on two architectures: convolutional DNNs (CDNNs) and long short-term memory (LSTM) recurrent neural networks (RNNs), which are less demanding in terms of computational resources due to the reduced amount of free parameters in comparison with other DNNs. Thus, they provide an alternative to classical i-vectors, achieving comparable results, especially when dealing with short utterances.

Second, we explore one of the most prominent applications of DNNs in speech processing: as feature extractors. Here, DNNs are used to obtain a frame-by-frame representation of the speech signal, the so-called *bottleneck feature* (BNF) vector, which is learned directly by the network and is then used instead of traditional acoustic features as input in LID and SID systems based on i-vectors. This approach revolutionized these two fields, since they highly outperformed state-of-the-art systems (i-vector based on acoustic features). Our analysis focuses on how different configurations of the DNN used as BNF extractor, which is trained for ASR, influences performance of resulting features for LID and SID.

Finally, we propose a novel approach for LID, in which the DNN is used to extract a fixed-length utterance-level representation of speech segments known as *embedding*, comparable to i-vector, and overcoming the disadvantage of variable length sequence of BNFs. This embedding-based approach has recently shown promising results for SID, and our proposed system was able to outperform a strong state-of-the-art reference i-vector system on the last challenging 2015 and 2017 NIST LREs. We explore different architectures and data augmentation techniques to improve results of our system, obtaining comparable or better results than the well-established i-vectors.

This work was funded by the PhD scholarship Ayudas para contratos predoctorales para la formación de doctores 2013 (BES-2013-064886), under the project CMC-V2: Caracterización, Modelado y Compensación de Variabilidad en la Señal de Voz (TEC2012-37585-C02-01), funded by Ministerio de Economía y Competitividad, Spain.

2. DNN as a Classifier for LID

We call *end-to-end* DNN-based systems to those that perform the target task from the input, without any other backend. For LID, they usually take some input features and are trained to classify each input frame into one of the target languages, outputting the probability vector of a frame belonging to each language. Here, we use CDNNs and LSTMs, which have less parameters than other DNNs. Besides, LSTMs have shown to be a good model for time-depending sequences [2]. We present experiments with both architectures on a balanced subset of 8 languages from LRE 2009, on the 3 s task. We compare our systems to an i-vector baseline, with 1024-dimensional UBM, 400-dimensional i-vectors and cosine scoring.

2.1. Convolutional DNN for Language Recognition

CDNNs usually consist of convolution and subsampling layers [3]. The former aims to perform feature extraction and each of its units is connected to a local subset of units in the previous layer. Groups of these units share their parameters and form a *feature map* that extracts the same features from different locations in the input. The subsampling layer selects the maximum activation of each region on the input (*max-pooling*). The general scheme of our CDNN-LID system is depicted in Figure 1.

We use as input 56-dimensional MFCC-SDC [4] feature vectors and segments of 3 s. All the architectures have 3 hidden layers and we vary the number of filters (feature maps) in each one, keeping fixed their shape and max-pooling regions. The output layer is a fully-connected layer with softmax activation, which outputs the probability that a test segment belongs to a certain language. The network is trained with stochastic gradient descent to minimize the negative log-likelihood. We conducted experiments to evaluate the influence of differ-

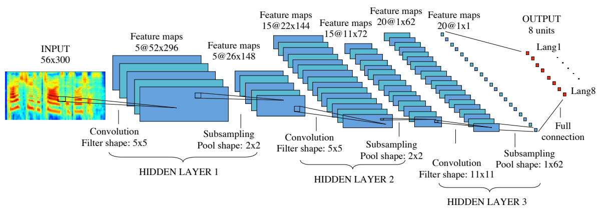

Figure 1: *Representation of a CDNN architecture for LID.*

Table 1: *Configuration of the CDNN models for LID.*

	Configuration	**Development Data**	
ID	# Filters/Layer	Train	Validation
ConvNet 1	[20, 30, 50]	∼178h	∼31h
ConvNet 2	[5, 15, 20]	∼178h	∼31h
ConvNet 3	[5, 15, 20]	∼356h	∼63h
ConvNet 4	[5, 15, 20]	∼534h	∼63h
ConvNet 5	[10, 20, 30]	∼356h	∼63h
ConvNet 6	[10, 20, 30]	∼534h	∼63h

Table 2: *CDNN systems performance for LID on the 8 languages subset of NIST LRE 2009.*

		Performance	
ID	Size	EER$_{avg}$ (%)	C_{avg}
i-vector	∼23M	**16.94**	**0.1535**
ConvNet 1	∼198k	22.14	0.2406
ConvNet 2	∼39k	25.90	0.2700
ConvNet 3	∼39k	24.69	0.2616
ConvNet 4	∼39k	23.48	0.2461
ConvNet 5	∼78k	21.60	0.2282
ConvNet 6	∼78k	21.11	0.2293
ConvNet 6+i-vector	-	**15.96**	**0.1433**

ent amounts of data used to train, balanced per language. The configurations used are summarized in Table 1.

2.1.1. *Experiments and Results*

Results of this section are summarized in Table 2. Although our standalone CDNN-based systems are outperformed by the i-vector, their size is smaller and are trained with less data. We see that systems benefit from larger training datasets (compare systems 2, 3 and 4) and bigger models (compare ConvNet 3 and 5, or 4 and 6). Moreover, improvements are obtained when fusing the best CDNN system with the i-vector baseline, meaning complementary information extracted from the same features.

2.2. LSTM RNN for Language Identification

LSTMs are able to store information from previous inputs during long time periods [5, 6, 7], which makes them more suitable to model sequential data. They replace hidden units in a classical DNN with *memory blocks* [8], which have input, output and forget gates: the input and output gates control respectively the flow of input activations into the memory cell and the output flow of cell activations into the rest of the network; the forget gate allows the flow of information from the memory block to the cell, adaptively resetting the cell's memory.

Table 3: *LSTM systems performance for LID on the 8 languages subset of NIST LRE 2009.*

Architecture			**Performance (%)**		
ID		Size	Acc.	EER$_{avg}$	Impr.
#1	i-vector	23M	65.02	16.94	-
#2	lstm_1×512	1.2M	57.51	17.82	-
#3	lstm_1×750	2.5M	63.39	15.61	∼7.85
#4	lstm_1×1024	4.4M	65.63	15.10	∼10.86
#5	lstm_2×256	850k	63.73	14.96	∼11.69
#6	lstm_2×512	3.3M	**70.90**	**12.51**	∼**26.15**

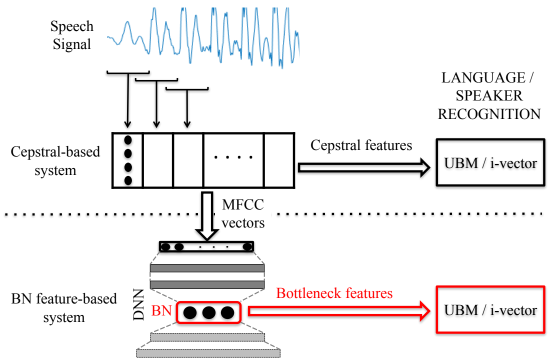

Figure 2: *Scheme of cepstral vs. BN based LID/SID systems.*

We train our systems with sequences of 2 s of MFCC-SDCs with no stacking of acoustic frames. They consist of 1 or 2 LSTM hidden layers followed by a softmax output layer, which returns a probability for each input frame and language. For scoring, we average the output per frame, using just the last 10% of each utterance.

2.2.1. *Experiments and Results*

Table 3 summarizes the performance of 5 LSTM systems in terms of EER$_{avg}$ and accuracy. We see that 4 out of the 5 proposed architectures for the LSTM system outperform the reference i-vector based system in EER$_{avg}$, with 5 to 21 times fewer parameters. In the models with one layer, increasing the number of units improves the performance. However, deeper models (2 layers) yield better results.

3. Frame-by-frame DNN-based Representation: Bottleneck Features

Generally, LID and SID systems based on BNF use a DNN with a bottleneck (BN) layer that is trained for ASR. Then, for each input frame, the time-dependent output of the BN layer is used as a new frame-by-frame representation to feed the i-vector model, instead of the classical cepstral features (see Figure 2).

Thus, BNFs provide a new frame-wise representation of an audio signal, learned directly by a DNN, containing information about the phonetic content since the DNN is trained for ASR. The BN layer of this DNN is relatively small with respect to the rest and aims to compress the information learned by the previous layers [9].

Table 4: *Results of BNFs for LID (development set of NIST LRE 2015), varying the number of layers of the DNN.*

Number of Hidden Layers	DNN Frame Acc.	EER$_{avg}$ (in %) 30s	10s	3s
3	47.82	5.52	9.04	14.34
4	49.55	**4.33**	**7.81**	**13.76**
5	**50.46**	5.22	8.57	14.15

Table 5: *Results of BNFs for LID (development set of NIST LRE 2015) with different position for the BN layer.*

Position of BN Layer	DNN Frame Accuracy	EER$_{avg}$ (in %) 30s	10s	3s
First	49.17	9.37	12.24	16.59
Second	49.46	6.27	9.55	14.58
Third	**49.55**	**4.33**	**7.81**	**13.76**
Fourth	48.05	4.64	8.00	14.17

3.1. Analysis of Bottleneck Features for LID

Here, we analyze how the topology of the DNN trained for ASR influences the performance of the resulting BNFs for LID on the NIST LRE 2015 development dataset. We use a feedforward DNN with an input layer, three to five hidden layers, and the output layer. To feed the network, we use 20 MFCCs preprocessed with a context of 31 frames. The hidden layers are composed of 1500 units and the BN layer, of 80. The softmax output layer provides the probability of each input to correspond to a given phoneme state (3083 triphone states are used). We use stochastic gradient descent to optimize the cross-entropy.

3.1.1. Experiments and Results

First we vary the number of layers in the DNN from 3 to 5 (Table 4). Despite the 5 layers configuration gives better performance in terms of frame accuracy, it is the architecture with 4 hidden layers the one that reaches the lowest EER$_{avg}$ for LID. Therefore, the discriminative task (ASR) is easier for the DNN when the classifier is more complex (5 layers DNN) and, thus, improves the frame accuracy. However, that network is not being forced to focus on obtaining a compact representation of the signal, which is then used for LID.

Then, keeping fixed the number of layers to 4, we explore how the LID system performs depending on the position that the BN layer occupies in the DNN, which correspond to different levels of extracted information closer or further from the phonetic information (output layer). Results can be seen in Table 5. The closer the BN layer to the input layer, the noisier the resulting representation would be, which might explain the drop in performance for the first and second layers with respect to the results of the last two layers. The best performance in terms of EER$_{avg}$ for LID is obtained when the bottleneck layer is located in the third layer, but that result is very close to the one obtained with the BN at the fourth layer. Performance of the DNN also drops when the BN layer moves from layer third to fourth. In this topology, the BN layer in position fourth is connected directly to the output layer, resulting in a weight matrix that connects a small layer with just 80 hidden units with the output layer, of size 3083. These weights might be difficult to learn, which may explain this drop in performance of the DNN.

Table 6: *Results of BNFs for SID on the NIST SRE 2010.*

Features	Norm.	Phone Acc(%)	EER(%)
ASR feat.	Utt. CMN	**49.8**	2.51
MFCC$_{\Delta+\Delta\Delta}$	ST-CMVN	49.6	1.99
MFCC$_{20dim}$	ST-CMVN	45.57	**1.67**

3.2. Analysis of Bottleneck Features for SID

We explore whether DNNs suboptimal for ASR can provide better BNFs for SID. We present here experiments with different features to feed the DNN, either optimized for ASR ("ASR feat.") or for SID ("MFCC"). The ASR optimized features [10] are composed of 24 Mel-filter bank log outputs concatenated with 13 fundamental frequency (F0) features, with utterance mean normalization, which is what we used as default for ASR [11]. SID optimized features are the classical 20 MFCCs used for SID, either adding the derivatives or not, and normalized with short-term cepstral mean and variance normalization (ST-MVN). We evaluate the systems on the NIST SRE 2010, condition 5, female task [12].

3.2.1. Experiments and Results

The aspect analyzed in this section is the DNN input features, which are either optimized for ASR or SID ("ASR feat." vs. "MFCC"). Results of these experiments are summarized in Table 6.

We see that the ASR features (with per utterance mean normalization) yield better performance in terms of phone accuracy than the MFCCs since they are expected to be optimized for ASR. However, BNFs obtained from these DNNs do not seem to be as discriminative as the ones obtained with DNNs trained using MFCCs optimized for SID. Moreover, adding first and second derivatives to MFCCs provide better phone accuracy but resulted in a worse SID performance. We see as for LID that better ASR performance (in terms of phone accuracy) does not necessarily correspond to better SID performance.

4. Utterance Level Representation: DNN-based Embeddings

Despite the success of BNFs for SID [13, 14, 15] and LID [16, 17, 18, 19, 20, 21], the variable length of this frame-wise representation poses a challenge in consequent modeling. The classical *i-vector* compacts the utterance representation in a fixed-length vector. However, the aim of i-vectors is to capture information about sources of variability in the training data, but this information is not necessarily relevant to the target task.

In this section we use *embeddings* for LID (after their success for SID [22]), which are a fixed-length representation of an utterance extracted from a sequence summarizing DNN trained discriminatively for the target task (LID).

The DNN consists of a first part that works on a frame-by-frame basis from a given sequence of feature vectors, followed by a pooling layer, which in our case computes the mean and standard deviation over time of the activations of the previous layer. Finally, a number of hidden layers follow to capture the information contained in the input, providing a single vector of values per sequence (embeddings), which can be modeled by some other backend.

In particular, our DNN-embedding system takes stacked BNFs as input and use bidirectional LSTM (BLSTM) layers for

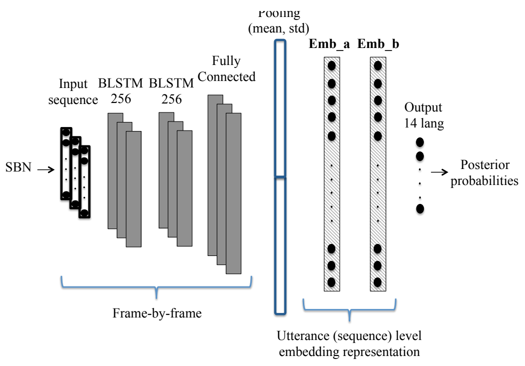

Figure 3: *Architecture of the proposed embedding DNN for LID.*

Table 7: *Comparison i-vectors vs. embeddings of different size for LID on the NIST LRE 2015 evaluation dataset.*

System	$C_{avg} \times 100$
Reference i-vector	**16.93**
DNN_1 (812)	20.04
DNN_2 (406)	19.19
DNN_3 (203)	19.30

the frame-level part. After pooling, two more fully connected layers are added, whose output values will serve as embeddings. Finally, the output layer consists of a softmax layer that provides a vector of language posterior probabilities for each utterance. An example of this architecture is depicted in Figure 3.

4.1. Analysis on NIST LRE 2015

For these experiments, we use the architecture described above. We feed the DNN with 30-dimensional stacked BNFs and the output softmax layer provides a 20-dimensional vector of language posterior probabilities for each utterance. As reference, we use an i-vector system that consists of a 2048-dimensional UBM trained on the same BNFs and 600 dimensional i-vectors.

First, we experiment varying the size of the embedding layers (keeping fixed the rest to 256 for each layer up to the pooling). We start with 512 and 300-dimensional embeddings (DNN_1) and half each twice (DNN_2 and 3, respectively). Results stacking both embeddings are shown in Table 7. We see a better performance of DNN_2 embeddings, which are half size w.r.t. DNN_1. This suggest that the embeddings of larger size contain more detrimental information about channel since all DNNs reached the same performance on the training data.

Motivated by this, we explore further dimensionality reduction via PCA in Table 8. We see that with smaller embeddings, we are able to get improvements even reducing the dimensionality up to 25. Best results are achieved with embeddings from DNN_2 whose dimensionality (406) is close to the typical i-vector (400 or 600). Besides, we achieved a performance of 17.44%, close to our i-vector baseline (16.93%) and score level fusion of both gave us a C_{avg} of 15.69%.

4.2. Analysis on NIST LRE 2017

After developing the embedding system for the NIST LRE 2017, where it was included in the primary submission of BUT team (a fusion of 3 i-vector systems and the embedding system),

Table 8: *Results with PCA on top of embeddings for LID on the NIST LRE 2015 evaluation dataset.*

	$C_{avg} \times 100$		
System	None	100	25
DNN_1 (orig 812)	20.04	18.67	19.98
DNN_2 (orig 406)	19.19	18.11	**17.44**
DNN_3 (orig 203)	19.30	18.70	18.13

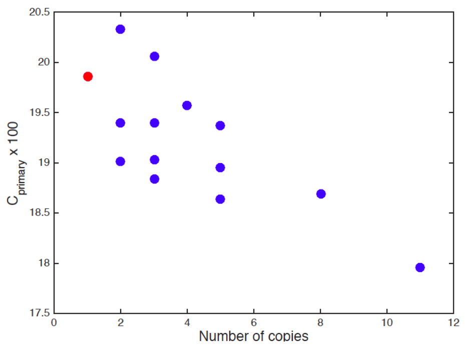

Figure 4: *Influence of data augmentation on DNN-embeddings for LID on the NIST LRE 2017 evaluation dataset.*

we performed a post-evaluation analysis.

First, we compared the architecture with the same configuration as DNN_1 in previous section with a larger one, where the fully connected layer has 1500 units and both embeddings are 512-dimensional. With that larger model, performance improved from 22.18% to 19.86% ($C_{primary}$).

Moreover, we extended the training dataset by performing data augmentation through addition of noise, reverberation and tempo variations of original audio files. Figure 4 shows the comparison of performance when training with up to 11 copies of the original data with different corruptions. In general, increasing the number of copies of the data yields improvements in performance. In particular, adding any noisy version of the data (combined or not with other corruptions) makes the system more robust against data mismatch, providing gains in performance. The only two cases in which data augmentation does not improve the system trained only on original data are the ones in which just reverberation or tempo variations are performed.

5. Conclusions

The main contributions of this Ph.D. Thesis are the following. First, the proposed end-to-end approaches for LID based on CDNNs and LSTMs, which provide an alternative to i-vectors with less parameters. Secondly, the systematic study of bottleneck feature DNN-based LID systems and the analysis of this approach for SID, which show that optimal DNN configuration for BNFs for LID and SID might differ from the most beneficial for ASR, task for which the DNN is trained. Finally, the novel approach based on embeddings for LID, in line with previous works in SID, which provides a fixed-length representation of utterances directly learned by the DNN for the target task able to outperform the well-established i-vectors.

In terms of articles, from research directly output from this Ph.D. Thesis, two journal articles and seven peer reviewed international conference papers were published.

6. References

[1] A. Lozano-Diez, “Bottleneck and embedding representation of speech for dnn-based language and speaker recognition,” Ph.D. dissertation, June 2018. [Online]. Available: https://repositorio.uam.es/handle/10486/684191

[2] T. Mikolov, S. Kombrink, L. Burget, J. H. Cernocky, and S. Khudanpur, “Extensions of recurrent neural network language model,” in *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*. IEEE, 2011, pp. 5528–5531.

[3] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” in *Intelligent Signal Processing*. IEEE Press, 2001, pp. 306–351.

[4] P. A. Torres-Carrasquillo, E. Singer, M. A. Kohler, and J. R. Deller, “Approaches to Language Identification Using Gaussian Mixture Models and Shifted Delta Cepstral Features,” in *Proc. ICSLP*, vol. 1, 2002, pp. 89–92.

[5] A. Graves, *Supervised Sequence Labelling with Recurrent Neural Networks*. Springer, 2012, vol. 385. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-24797-2

[6] F. A. Gers, J. Schmidhuber, and F. Cummins, “Learning to forget: Continual prediction with LSTM,” *Neural Computation*, vol. 12, no. 10, pp. 2451–2471, 2000.

[7] F. A. Gers, N. N. Schraudolph, and J. Schmidhuber, “Learning precise timing with LSTM recurrent networks,” *Journal of Machine Learning Research*, vol. 3, pp. 115–143, Mar. 2003.

[8] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber, “Lstm: A search space odyssey,” *arXiv preprint arXiv:1503.04069*, 2015.

[9] F. Grezl, M. Karafiat, S. Kontar, and J. Cernocky, “Probabilistic and bottle-neck features for lvcsr of meetings,” in *2007 IEEE International Conference on Acoustics, Speech and Signal Processing - ICASSP ’07*, vol. 4, April 2007, pp. IV–757–IV–760.

[10] F. Grézl, M. Karafiát, and L. Burget, “Investigation into bottleneck features for meeting speech recognition,” in *Proc. Interspeech 2009*, no. 9. International Speech Communication Association, 2009, pp. 2947–2950.

[11] M. Karafiát, F. Grézl, K. Veselý, M. Hannemann, I. Szőke, and J. Černocký, “But 2014 babel system: Analysis of adaptation in nn based systems,” in *Proceedings of Interspeech 2014*. International Speech Communication Association, 2014, pp. 3002–3006.

[12] NIST, “The nist year 2010 speaker recognition evaluation plan,” www.itl.nist.gov/iad/mig/tests/sre/2010/NIST_SRE10_evalplan.r6.pdf, 2010.

[13] D. Garcia-Romero and A. McCree, “Insights into deep neural networks for speaker recognition,” in *INTERSPEECH 2015, 16th Annual Conference of the International Speech Communication Association, Dresden, Germany, September 6-10, 2015*, 2015, pp. 1141–1145.

[14] S. Yaman, J. Pelecanos, and R. Sarikaya, “Bottleneck features for speaker recognition,” in *Proceedings of Odyssey 2012*. International Speech Communication Association, 2012.

[15] A. Lozano-Diez, A. Silnova, P. Matějka, O. Glembek, O. Plchot, J. Pešán, L. Burget, and J. Gonzalez-Rodriguez, “Analysis and optimization of bottleneck features for speaker recognition,” in *Proceedings of Odyssey 2016*. International Speech Communication Association, 2016.

[16] Y. Song, B. Jiang, Y. Bao, S. Wei, and L. R. Dai, “I-vector representation based on bottleneck features for language identification,” *Electronics Letters*, vol. 49, no. 24, pp. 1569–1570, November 2013.

[17] A. Lozano-Diez, R. Zazo, D. T. Toledano, and J. Gonzalez-Rodriguez, “An analysis of the influence of deep neural network (dnn) topology in bottleneck feature based language recognition,” *PLOS ONE*, vol. 12, no. 8, pp. 1–22, 08 2017. [Online]. Available: https://doi.org/10.1371/journal.pone.0182580

[18] R. Fér, P. Matějka, F. Grézl, O. Plchot, and J. Černocký, “Multilingual bottleneck features for language recognition,” in *Proceedings of Interspeech 2015*, vol. 2015, no. 09, 2015, pp. 389–393.

[19] P. Matějka, L. Zhang, T. Ng, H. S. Mallidi, O. Glembek, J. Ma, and B. Zhang, “Neural network bottleneck features for language identification,” in *Proceedings of Odyssey 2014*. International Speech Communication Association, 2014.

[20] F. Richardson, D. Reynolds, and N. Dehak, “Deep neural network approaches to speaker and language recognition,” *IEEE Signal Processing Letters*, vol. 22, no. 10, pp. 1671–1675, Oct 2015.

[21] B. Jiang, Y. Song, S. Wei, J.-H. Liu, I. V. McLoughlin, and L.-R. Dai, “Deep bottleneck features for spoken language identification,” *PLOS ONE*, vol. 9, no. 7, pp. 1–11, 07 2014. [Online]. Available: https://doi.org/10.1371/journal.pone.0100795

[22] D. Snyder, D. Garcia-Romero, D. Povey, and S. Khudanpur, “Deep neural network embeddings for text-independent speaker verification,” in *Proceedings of Interspeech 2017*, 2017.